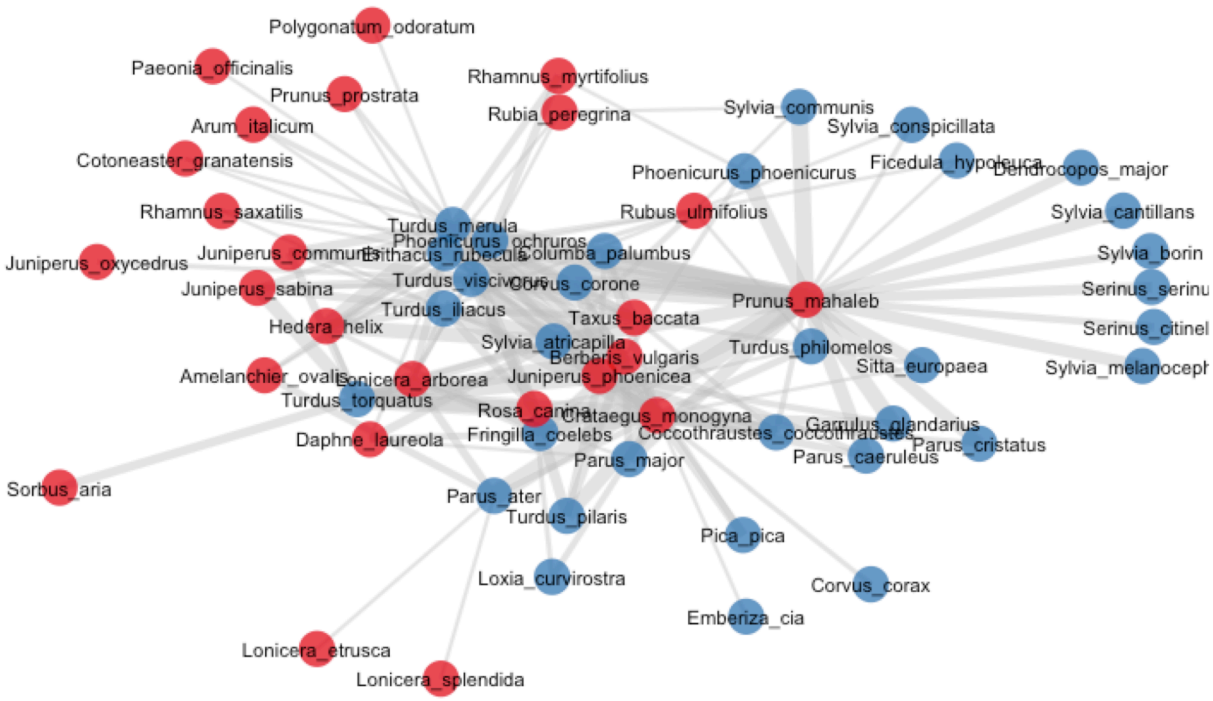
télécom saint-étienne
école d'ingénieurs nouvelles technologies

Rapport de Project EGC

Etude et analyse de la relation entre les documents



ZunzunWANG

YishuoLYU

École associée INSTITUT Mines-Télécom

Université Jean Monnet SAINT-ETIENNE

Sommaire

1 - Introduction

Le Data Mining est en fait un terme générique englobant toute une famille d'outils facilitant l'exploration et l'analyse des données contenues au sein d'une base décisionnelle de type Data Warehouse ou DataMart. Les techniques mises en action lors de l'utilisation de cet instrument d'analyse et de prospection sont particulièrement efficaces pour extraire des informations significatives depuis de grandes quantités de données.

Avec le développement des outilles Data Mining, les outilles permettant d'extraire une valeur dans ces données se sont multipliés. Nous avons utilisé premièrement le text mining pour prétraitement les documents. Et ensuite on utilise le social mining pour transférer notre donnée à un graphe des documents.

On voudra automatiser le traitement de gros volumes de contenus texte pour en extraire les principales tendances et répertorier de manière statistique les différents sujets évoqués.

2 - Présentation de donnée

Les données misent à notre disposition décrivent les 1041 articles qui ont été présentés à la conférence EGC depuis 2014.

Chaque enregistrement, correspondant à un article, est décrit par 8 champs :

- ***series***
  ex: Revue des Nouvelles Technologies de l'Information
- ***booktitle***
  ex: EGC
- ***year***
  ex :2008-2015
- ***title***
  ex : A Clustering Based Approach for Type Discovery in RDF Data Sources
- ***abstract***
  ex :RDF(S)/OWL data sources are not organized according to a predefined schema, as they are structure less by nature. This lack of schema limits their use to express queries or to understand their content. Our work is a contribution towards the inference of the structure of RDF(S)/OWL data sources. We present an approach relying on density-based clustering to discover the types describing the entities of possibly incomplete and noisy data sets.
- ***author***
  ex : Kenza Kellou-Menouer, Zoubida Kedad
- ***pdf1page***
  ex :http://editions-rnti.fr/render_pdf.php?p1&p=1002113
- ***pdfarticle***
  ex : http://editions-rnti.fr/render_pdf.php?p=1001177

- Sur les 1041 articles seuls, 896 ont un résumé.

3 - *Objectifs*

A travers cette étude, nous avons analysé les données à notre disposition afin de déterminer le mot clé de chaque article, ainsi que la relation entre différents articles. Nous avons aussi pu voir les changements quand on modifie le règle pour définir la relation entre différents articles (C'est à dire que le quantité de mot commun entre les différents articles décide la relation).

4 - *Graph de Processus*

Prétraitement de donnée

- Cette première étape nous a permis d'avoir une vue générale des différents attribus. Après on choisit l'abstract comme notre attribut pour analyser.

Text mining

- traitement selon chaque document.
- trouvez le mot clé de chaque article.

Social mining

- création les graphe des documents
  1) Bipartite graph: relation entre chaque abstract et mot clé.
- 2) Directed graph, undirected graph, weighted graph
- 3) Analyse les graphs

5 - *Prétraitement*

Après l'analyse en détail de ce fichier, nous trouvons que les données contenues dans le champ 'abstract' inclus plus de information par la description plus courts et ceci est plus facile pour trouver la relation entre eux.

Compte tenu de ces données plus représentatives, donc nous décidons de choisi ces données pour les analyses suivantes principales.

6 - *Text Mining*

Ici, nous faisons de l'analyse sur la relation des articles par leur propre abstract.

La processus de l'analyse est constitué par **text mining** et **social mining** .

D'abord, nous faisons la partie **text mining**, et les suivants sont les codes de R sur la partie de **text mining**, c'est-à-dire la prétraitement des données.

```r
#text mining
Sys.setenv(JAVA_HOME='D:/Program Files (x86)/jdk1.7/jre')
library(XLConnect)
library(tm)
library(RColorBrewer)
#vignette("tm")

options(max.print=1000000)
cas<-readWorksheet(loadWorkbook("C:/Users/lv/Desktop/Data mining/R_Projet/Cas2015_16.xlsx"),sheet=1)
cas
abstract <- cas[1:1041,5]
reuters<-Corpus(VectorSource(abstract))
reuters <- tm_map(reuters, PlainTextDocument)
reuters <- tm_map(reuters, stripWhitespace)
reuters <- tm_map(reuters, removePunctuation)
reuters <- tm_map(reuters, removeNumbers)
reuters <- tm_map(reuters, content_transformer(tolower))
reuters <- tm_map(reuters, removeWords, stopwords("english"))
reuters <- tm_map(reuters, removeWords, stopwords("french"))
reuters <- tm_map(reuters, removeWords,
                  c("afin","ainsi","alors","according","comme","dun","dune","entre",
                    "ensemble","¨être","non","très"))

tm_map(reuters, stemDocument)

dtm <- DocumentTermMatrix(reuters)

#The value of 'sparse' contribute to the terms left.
dtm2 <- removeSparseTerms(dtm, sparse=0.99)
data <- as.data.frame(inspect(dtm2))

findFreqTerms(dtm2, 250)
findAssocs(dtm,"lagrange",0.3)

name <- matrix(1:1041,1041,1)
rownames(data)<-name[,1]

write.table(data,"C:/Users/lv/Desktop/Data mining/R_Projet/result.csv",sep=",")
```

(1) Il faut importer les packages nécessaires et configurer le path de jre.

```
Sys.setenv(JAVA_HOME='D:/Program Files (x86)/jdk1.7/jre')
library(XLConnect)
library(tm)
library(RColorBrewer)
#vignette("tm")
```

(La commande de vignette ('tm') est pour regarder le document de la package 'tm'.)

(2) Nous définissons les records maximals présentés, et après nous importons les données nécessaires 'Cas2015_16.xlsx' et récupérer seulement les données contenues dans le champ 'abstract'.

```
options(max.print=1000000)
cas<-readWorksheet(loadWorkbook("C:/Users/lv/Desktop/Data mining/R_Projet/Cas2015_16.xlsx"),sheet=1)
cas
abstract <- cas[1:1041,5]
```

(3) La création d'un corpus par un vecteur

```
abstract <- cas[1:1041,5]
```

(4) Nous prétraiter les données contenues dans le champ 'abstract', parce que ils n'ont pas été nettoyées (espaces manquants, tirets de mesure coupant à tort certains mots etc.).

```
reuters <- tm_map(reuters, PlainTextDocument)
reuters <- tm_map(reuters, stripWhitespace)
reuters <- tm_map(reuters, removePunctuation)
reuters <- tm_map(reuters, removeNumbers)
reuters <- tm_map(reuters, content_transformer(tolower))
reuters <- tm_map(reuters, removeWords, stopwords("english"))
reuters <- tm_map(reuters, removeWords, stopwords("french"))
reuters <- tm_map(reuters, removeWords,
                  c("afin","ainsi","alors","according","comme","dun","dune","entre",
                    "ensemble","^être","non","très"))
tm_map(reuters, stemDocument)
```

-> PlainTextDocument : La transformation de documents XML en texte.
-> stripWhitespace : L'élimination des espaces supplémentaires.
-> removePunctuation : L'élimination des ponctuations
-> removeNumbers : L'élimination des nombres.
-> tolower : La conversion en minuscule.
-> removeWords, stopwords("english") : L'élimination des mots d'arrêt.
-> removeWords, stopwords("french") : L'élimination des mots d'arrêt.
-> removeWords, c("afin","ainsi"…) : La création d'une dictionnaire et l'éliminer.
-> stemDocument : Stemming.

(5) La création d'un matrice dtm par reuters.

```
dtm <- DocumentTermMatrix(reuters)
```

(6) L'élimination des termes clairsemées et la conversion en matrice norme.

```
dtm2 <- removeSparseTerms(dtm, sparse=0.99)
data <- as.data.frame(inspect(dtm2))
```

Le résultat de dtm2 est :

```
> dtm2
<<DocumentTermMatrix (documents: 1041, terms: 880)>>
Non-/sparse entries: 26456/889624
Sparsity           : 97%
Maximal term length: 20
Weighting          : term frequency (tf)
```

(7) La recherche de termes ayant au moins 250 occurrences.

```
32  findFreqTerms(dtm2, 250)
```

Le résultat est :

```
> findFreqTerms(dtm2, 250)
[1] "approche"      "article"       "classification" "données"       "méthode"       "plus"
[7] "proposons"
```

(8) La recherche de mots ayant un taux de corrélation supérieure ou égale à 0.3 avec le mot de ‘lagrange’.

```
33  findAssocs(dtm,"lagrange",0.3)
```

Le resultat est :

```
> findAssocs(dtm,"lagrange",0.3)
$lagrange
multiplicateurs      transformé        spectrale         établit   relationnelle          résolu
           1.00            1.00             0.87            0.71            0.67            0.58
          trace   catégorielles      relaxation       problème         propres
           0.58            0.50             0.50            0.37            0.33
```

(9) Nous changeons les noms contenus dans le premier rang (les noms de articles) à cause de noms pareils après le prétraitement des données.

```
35  name <- matrix(1:1041,1041,1)
36  rownames(data)<-name[,1]
```

(10) Emporter les résultats au le fichier .csv

```
38  write.table(data,"C:/Users/lv/Desktop/Data mining/R_Projet/result.csv",sep=",")
```

(11) Les mots-clés dans le ‘data’ que nous avons recherché sont comme :

	Terms										
Docs	caractériser	caractéristiques	carte	cartes	cas	catégories	celle	celles	cellesci	celui	cependant
character(0)	0	0	0	0	0	0	0	0	0	0	0
character(0)	0	0	0	0	0	0	0	0	0	0	1
character(0)	0	0	0	0	0	0	0	0	0	0	0
character(0)	0	0	0	0	0	0	0	0	0	0	0
character(0)	0	0	0	0	1	0	0	0	0	0	0

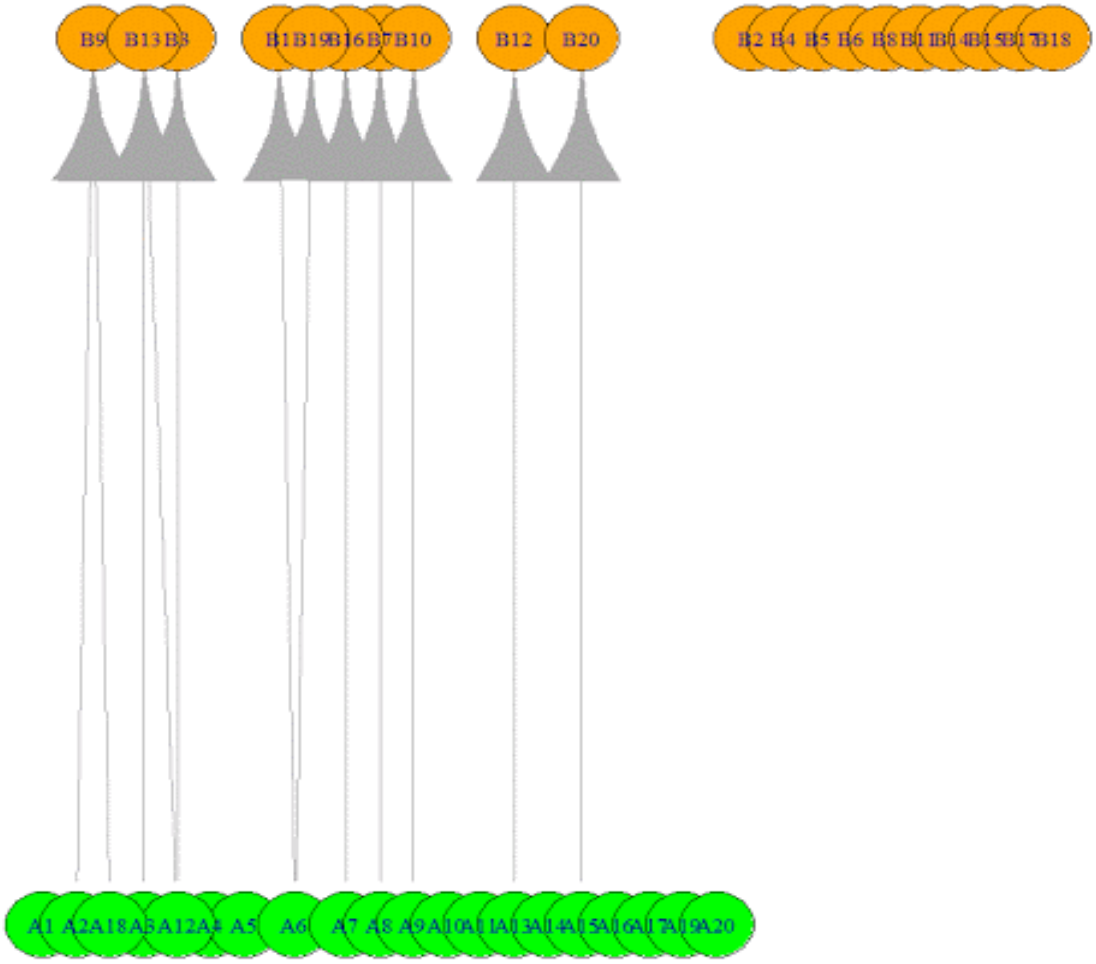
7 - Social Mining

Ensuite, après le prétraitement des données et text mining, nous allons faire la partie 'social mining'.

D'abord, nous intégrons les mots-clés de tous les articles comme un dictionnaire, et après nous pouvons obtenir la fréquence des mots-clés de chaque article dans ce dictionnaire. Enfin nous pouvons diviser les articles selon ces différentes fréquences. (On peut saisir s'il y a une relation entre les deux articles par les différents seuils.)

```
Sys.setenv(JAVA_HOME='D:/Program Files (x86)/jdk1.7/jre')
library(XLConnect)
library(igraph)
Cas <- read.csv("C:/Users/lv/Desktop/Data mining/R_Projet/result.csv")
nodesSet1 <- 1:20
nodesSet2 <- 1:20
indexH <- 1
ss1 <-diag(0, nrow=1, ncol=12)
ss2 <-diag(0, nrow=1, ncol=12)
for (i in seq(from=1,to=20)){
  for(n in seq(from=1, to=20)){
    if(Cas[i,n]!=0){
      ss1[1,indexH] <- i
      ss2[1,indexH] <- n
      indexH <- indexH + 1
    }
  }
}
# create example input data
edgeList <- data.frame(S1=c(ss1),S2=c(ss2))

### PREMISE :
### graph.bipartite function simply create a graph and add a boolean 'type' attribute
### to each vertex that indicate the partition of the vertex (e.g. TRUE first partition,
### FALSE second partition).
### So, it's not strictly necessary to use that function to get a bipartite graph, but
### you can use any method you like (or feel easier) as long as you add a 'type'
### attribute.
### Hence, in the following code I won't use graph.bipartite since I don't like it :)

# first we give prefixes to the nodes to discern the two partition
g <- graph.empty()
g <- add.vertices(g,nv=length(nodesSet1),attr=list(name=paste0('A',nodesSet1),
                                                   type=rep(TRUE,length(nodesSet1))))
g <- add.vertices(g,nv=length(nodesSet2),attr=list(name=paste0('B',nodesSet2),
                                                   type=rep(FALSE,length(nodesSet2))))
# we need to turn edgeList into a vector (and using names instead of indexes)
edgeListVec <- as.vector(t(as.matrix(data.frame(S1=paste0('A',edgeList$S1),
                                                S2=paste0('B',edgeList$S2)))))
g <- add.edges(g,edgeListVec)
# check if is recognized as bipartite
is.bipartite(g)
plot.igraph(g, layout=layout.bipartite,
            vertex.color=c("orange","green")[V(g)$type+1])
```

Comme ci-dessous, les ronds verts représentent les différents documents et les ronds jaunes représentent les mots qui référencent dans le dictionnaire.



Ensuit on va définir le relation entre les différent documents par le seuil.

```
Sys.setenv(JAVA_HOME='D:/Program Files (x86)/jdk1.7/jre')
library(XLConnect)
Cas <- read.csv("C:/Users/lv/Desktop/Data mining/R_Projet/result.csv")

table <- diag(0, nrow=20, ncol=20)
tableno <- diag(0, nrow=20, ncol=20)

#row numero
for (i in seq(from=1,to=19)){
  # row numero compare
  for(n in seq(from=i+1, to=20)){
    contenu=0
    #column numero
    for (j in 1:length(Cas[1,])){
      x <- Cas[i,j]
      y <- Cas[n,j]
      if (x&y == TRUE){
        contenu <- contenu+1
      }
    }
#    print (contenu)
    if(contenu>3){
      table[i,n] <- 1
      table[n,i] <- 1
      tableno[i,n] <- 1
    }
  }
}

write.table(table,"C:/Users/lv/Desktop/Data mining/R_Projet/table.csv",sep=",")
write.table(tableno,"C:/Users/lv/Desktop/Data mining/R_Projet/tableno.csv",sep=",")
```

(1) L'initialisation des deux matrices 20*20. La matrice 'table' est pour sauvegarder les données pour faire le graph avec direction et la matrice 'tableno' est pour sauvegarder les données pour faire le graph non direction.

```
table <- diag(0, nrow=20, ncol=20)
tableno <- diag(0, nrow=20, ncol=20)
```

(2) Les cycles 'for' suivants réalisent principalement :
Nous pensons que les deux articles ont une relation si ils ont plus de 3 mots pareils. Nous mettons le numéro des articles comme l'axe horizontal et vertical des nouvelles matrices. Si les deux articles ont une relation, nous pensons que sa valeur est 1, sinon est 0.

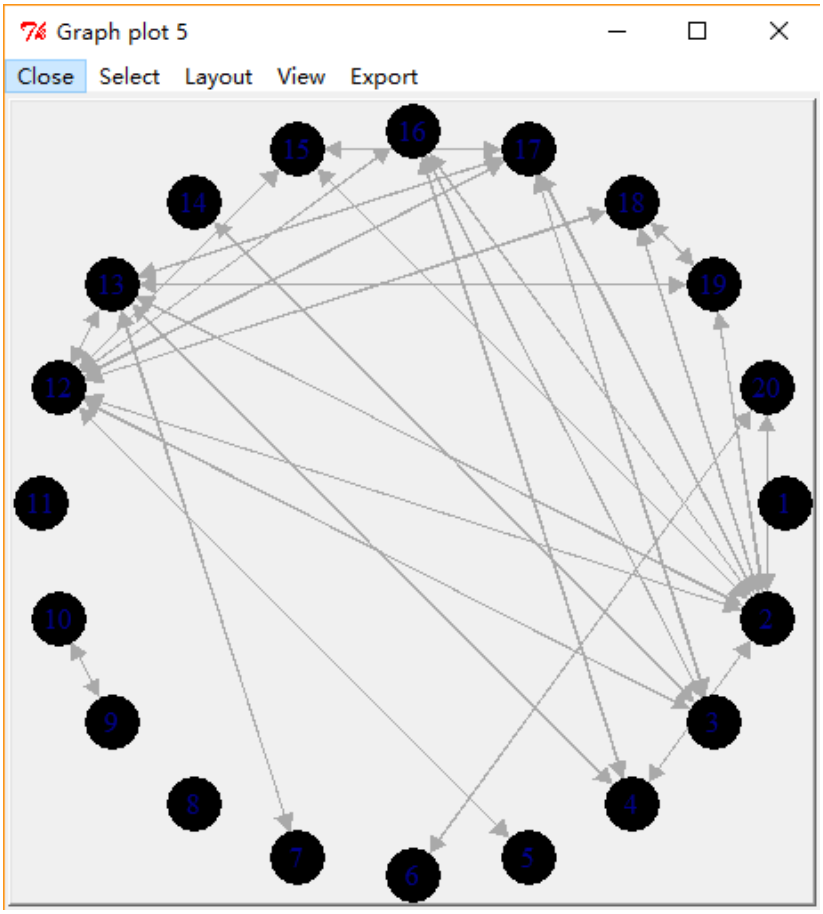
```
#row numero
for (i in seq(from=1,to=19)){
  # row numero compare
  for(n in seq(from=i+1, to=20)){
    contenu=0
    #column numero
    for (j in 1:length(Cas[1,])){
      x <- Cas[i,j]
      y <- Cas[n,j]
      if (x&y == TRUE){
        contenu <- contenu+1
      }
    }
#    print (contenu)
    if(contenu>3){
      table[i,n] <- 1
      table[n,i] <- 1
      tableno[i,n] <- 1
    }
  }
}
```

(3) Emporter séparément les résultats au les fichiers .csv

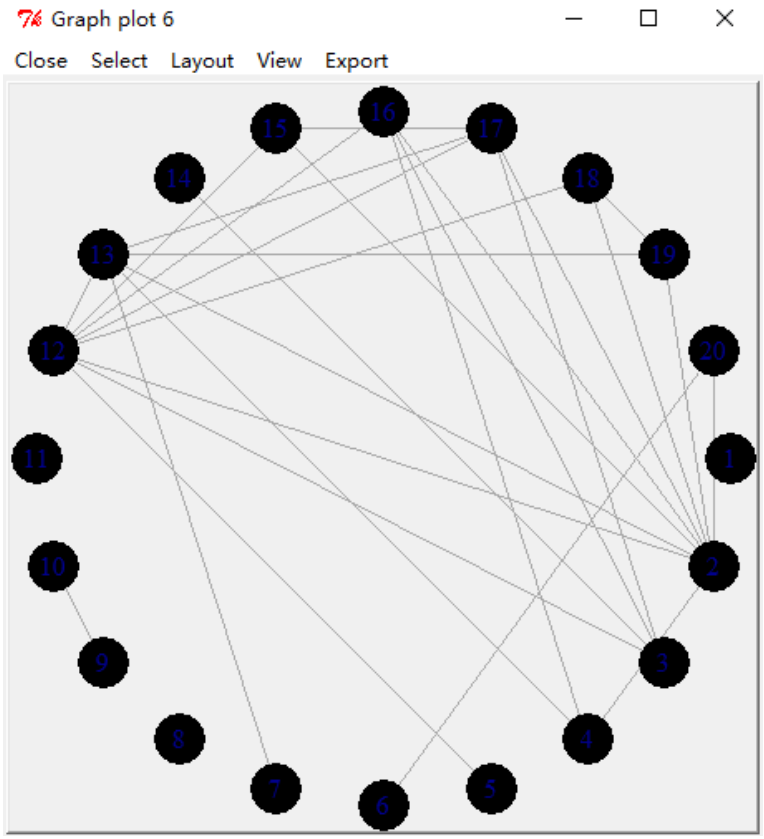
```
write.table(table,"C:/Users/lv/Desktop/Data mining/R_Projet/table.csv",sep=",")
write.table(tableno,"C:/Users/lv/Desktop/Data mining/R_Projet/tableno.csv",sep=",")
```

Ensuit on va analyser les données par le directed graph et undirected graph.

```
Sys.setenv(JAVA_HOME='D:/Program Files (x86)/jdk1.7/jre')
library(XLConnect)
library(igraph)
table <- read.csv("C:/Users/lv/Desktop/Data mining/R_Projet/table.csv")
tableno <- read.csv("C:/Users/lv/Desktop/Data mining/R_Projet/tableno.csv")

tableArc <- diag(0, nrow=56, ncol=2)
index <- 1
#row numero
for (i in seq(from=1,to=20)){
  # row numero compare
  for(n in seq(from=1, to=20)){
    if(table[i,n]==1){
      tableArc[index,1] <- i
      tableArc[index,2] <- n
      index<-index+1
    }
  }
}

tableArcno <- diag(0, nrow=28, ncol=2)
indexno <- 1
#row numero
for (i in seq(from=1,to=20)){
  # row numero compare
  for(n in seq(from=i+1, to=20)){
    if(tableno[i,n]==1){
      tableArcno[indexno,1] <- i
      tableArcno[indexno,2] <- n
      indexno<-indexno+1
    }
  }
}
tableArcno

#go2 <- graph(c(tableArc),n=20)
go2 <- graph(t(tableArc))
gno2 <- graph(t(tableArcno),directed = FALSE)

tkplot(go2)  #avec direction
tkplot(gno2) #non direction
```



directed graph



undirected graph

À la fin on analyse les dégrée

```
43  #the number of vertices
44  V(go2)
45
46  #the number of edges
47  E(go2)
48
49  #the degree of every vertice
50  degree(go2)
51
52  #the distribution of degrees of vertices
53  degree.distribution(go2)
54
55  #the shortest path of every vertices(the number of stop vertices)
56  shortest.paths(go2)
57
58  get.shortest.paths(go2,2)
59
60  #the similarity between vertices
61  similarity.jaccard(go2)
62
63  #the centrality
64  closeness(go2)
65
66  #the betweeness
67  betweenness(go2)
68  edge.betweenness(go2)
69
70  #the connexity
71  is.connected(go2)
72  clusters(go2)
73  no.clusters(go2)
74
75  #the communitty detection
76  wtcgo2 <- walktrap.community(go2)
77  ebc <- edge.betweenness.community(go2, directed = TRUE, edge.betweenness = TRUE, merges = TRUE, bridges = TRUE)
78  #display the communities
79  wtcgo2$membership
80  #modularity of the associated partition
81  modularity(wtcgo2)
82  modularity(go2,membership(wtcgo2))
83
84  #graphs generators
85  g <- erdos.renyi.game(1000, 1/1000)
86  g <- barabasi.game(10000)
87  #g <- graph.ring(10)as.directed(g, "mutual")
```

```
> V(go2)
+ 20/20 vertices:
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
> E(go2)
+ 56/56 edges:
 [1]  2-> 4  2->12  2->13  2->15  2->16  2->17  2->18  2->19  2->20  3->12  3->14  3->16  3->17  4-> 2  4->13
[16]  4->16  5->12  6->20  7->13  9->10 10-> 9 12-> 2 12-> 3 12-> 5 12->13 12->15 12->16 12->17 12->18 13-> 2
[31] 13-> 4 13-> 7 13->12 13->17 13->19 14-> 3 15-> 2 15->12 15->17 16-> 2 16-> 3 16-> 4 16->12 17-> 2 17-> 3
[46] 17->12 17->13 17->15 18-> 2 18->12 18->19 19-> 2 19->13 19->18 20-> 2 20-> 6
```

```
> degree(go2)
 [1]  0 18  8  6  2  2  2  0  2  2  0 16 12  2  6  8 10  6  6  4
```

```
> degree.distribution(go2)
 [1] 0.15 0.00 0.30 0.00 0.05 0.00 0.20 0.00 0.10 0.00 0.05 0.00 0.05 0.00 0.00 0.00 0.05 0.00 0.05
```

```
> shortest.paths(go2)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19]
 [1,]    0  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf  Inf
 [2,]  Inf    0    2    1    2    2    2  Inf  Inf  Inf  Inf    1    1    3    1    1    1    1    1
 [3,]  Inf    2    0    2    2    4    3  Inf  Inf  Inf  Inf    1    2    1    2    1    1    2    3
 [4,]  Inf    1    2    0    3    3    2  Inf  Inf  Inf  Inf    2    1    3    2    1    2    2    2
 [5,]  Inf    2    2    3    0    4    3  Inf  Inf  Inf  Inf    1    2    3    2    2    2    2    3
 [6,]  Inf    2    4    3    4    0    4  Inf  Inf  Inf  Inf    3    3    5    3    3    3    3    3
 [7,]  Inf    2    3    2    3    4    0  Inf  Inf  Inf  Inf    2    1    4    3    3    2    3    2
```

```
> get.shortest.paths(go2,2)
$vpath
$vpath[[1]]
+ 0/20 vertices:

$vpath[[2]]
+ 1/20 vertex:
[1] 2

$vpath[[3]]
+ 3/20 vertices:
[1]  2 12  3
```

```
> similarity.jaccard(go2)
      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7] [,8] [,9] [,10] [,11]      [,12]
 [1,]    1 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000    0    0     0     0 0.0000000
 [2,]    0 1.0000000 0.3000000 0.2000000 0.1111111 0.1111111 0.1111111    0    0     0     0 0.4166667
 [3,]    0 0.3000000 1.0000000 0.1666667 0.2500000 0.0000000 0.0000000    0    0     0     0 0.2000000
 [4,]    0 0.2000000 0.1666667 1.0000000 0.0000000 0.0000000 0.3333333    0    0     0     0 0.3750000
 [5,]    0 0.1111111 0.2500000 0.0000000 1.0000000 0.0000000 0.0000000    0    0     0     0 0.0000000
 [6,]    0 0.1111111 0.0000000 0.0000000 0.0000000 1.0000000 0.0000000    0    0     0     0 0.0000000
 [7,]    0 0.1111111 0.0000000 0.3333333 0.0000000 0.0000000 1.0000000    0    0     0     0 0.1250000
 [8,]    0 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000    1    0     0     0 0.0000000
 [9,]    0 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000    0    1     0     0 0.0000000
[10,]    0 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000    0    0     1     0 0.0000000
```

```
> closeness(go2)
 [1] 0.002631579 0.008333333 0.007751938 0.007812500 0.007462687 0.006944444 0.007299270 0.002631579
 [9] 0.002770083 0.002770083 0.002631579 0.008264463 0.008064516 0.007042254 0.007812500 0.007936508
[17] 0.008064516 0.007812500 0.007692308 0.007633588
```

```
> betweenness(go2)
 [1]  0.000000 72.333333 26.666667  1.333333  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
[11]  0.000000 50.666667 33.000000  0.000000  0.000000 10.000000 12.666667  2.000000  1.333333 26.000000
```

```
> edge.betweenness(go2)
 [1]  6.666667  9.666667  8.000000  6.666667  8.333333  7.166667  6.166667  7.666667 26.000000 12.000000
[11] 14.000000  6.666667  8.000000  6.666667  4.333333  4.333333 14.000000 14.000000 14.000000  1.000000
[21]  1.000000  9.666667 12.000000 14.000000  9.666667  4.666667  4.666667  2.833333  7.166667  8.000000
[31]  4.333333 14.000000  9.666667  6.000000  5.000000 14.000000  6.666667  4.666667  2.666667  8.333333
[41]  6.666667  4.333333  4.666667  7.166667  8.000000  2.833333  6.000000  2.666667  6.166667  7.166667
[51]  2.666667  7.666667  5.000000  2.666667 26.000000 14.000000
```

```
> is.connected(go2)
[1] FALSE
> clusters(go2)
$membership
 [1] 1 2 2 2 2 2 2 3 4 4 5 2 2 2 2 2 2 2 2 2

$csize
[1]  1 15  1  2  1

$no
[1] 5
```

```
> no.clusters(go2)
[1] 5
```

```
> modularity(wtcgo2)
[1] 0.2193878
```

```
> modularity(go2,membership(wtcgo2))
[1] 0.2193878
```

8 - *Conclusion générale*

A travers cette étude, nous avons utilisé différentes méthodes de data mining sur des différents documents.

Ces méthodes nous ont permis :

1. Trouver les mots clés dans chaque document.
2. Trouver les mots communs dans différents documents.
3. Regrouper les différents documents par la quantité de mot clé.
4. Trouver des similitudes dans les différentes abstracts.